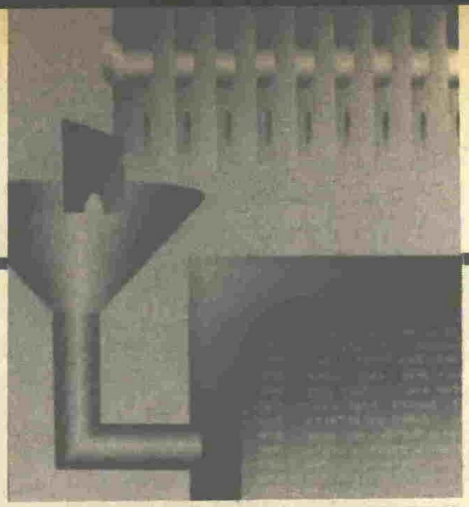

■ SOFTMAIL

การแปลงโฉมข้อมูล

ขจร โรจนเมธินทร์

ทำไมต้องแปลงโฉมด้วย

คุณคงจะสงสัยว่าการแปลงโฉมข้อมูลนี้หมายถึงอะไร อันที่จริงการแปลงโฉมข้อมูลก็คือการนำเอาข้อมูลที่เราสามารถอ่านออกมา แปลงด้วยวิธีใดวิธีหนึ่งเพื่อให้ข้อมูลนั้นอ่านไม่ออก คราวนี้บางคนอาจคิดอยู่ในใจแล้วว่าจำเป็นต้องแปลงด้วยหรือ ผมคงตอบว่าก็จำเป็นสำหรับข้อมูลที่มีความสำคัญมาก หรือข้อมูลที่เป็นความลับและไม่ต้องการให้คนธรรมดาอ่าน

ถ้าลองย้อนหลังกลับไปในอดีต การแปลงข้อมูลมีมาตั้งแต่สมัยกรีกและได้มีการพัฒนามาตลอด จนถึงสมัยที่มีการประดิษฐ์เครื่องส่งโทรเลขก็ได้มีการคิดรหัสมอส (Morse code) ขึ้นมาใช้ในการส่งข้อมูล รหัสมอสนี้จะใช้สัญลักษณ์สองชนิดคือ จุด และ ขีด (Dots and Dashes) แทนตัวอักษรซึ่งเป็นการนำเอาวิธีแทนข้อมูลแบบไบนารี่ (Binary representation) มาใช้เป็นครั้งแรก ต่อมาในสมัยสงครามโลกครั้งที่ 1 ได้มีการสร้างเครื่องมือสำหรับใส่รหัสข้อมูล (Code Machine) ที่ใช้ในการเข้ารหัส (Encoding) และการถอดรหัส (Decoding) ข้อมูลสำคัญก่อนที่จะมีการสื่อสารกัน แต่ก็มีการพยายามที่จะขโมยถอดรหัสข้อมูลของฝ่ายศัตรู ดังนั้นวิธีการเข้ารหัสข้อมูลจึงได้มีการพัฒนากันต่อมา

ในวงการของคอมพิวเตอร์การเข้ารหัสข้อมูลมีความสำคัญมากโดยเฉพาะในระบบคอมพิวเตอร์แบบมัลติยูสเซอร์ (Multiuser computers) ที่ต้องมีการรักษาความปลอดภัยของข้อมูลที่สำคัญ ซึ่งก็สามารถทำได้โดยนำไฟล์มาเข้ารหัสทำให้อ่านไม่ออก นอกจากนี้ก็อาจใช้ในการแปลงรหัสผ่าน (Password) ที่จะเข้าระบบให้อ่านไม่ออกเพื่อความปลอดภัยในการเก็บรหัสผ่านนั้น

วิธีการเข้ารหัสข้อมูล (Ciphers)

วิธีการที่นิยมใช้ในการเข้ารหัสข้อมูลนั้นแบ่งได้ 3 วิธีคือ

1. **Substitution** วิธีนี้หลักการก็คือเอาข้อมูลที่ต้องการเข้ารหัสมาทีละอักขระแล้วแทนด้วยอักขระตัวใหม่ ทำให้ข้อมูลที่ได้ออกมาอ่านไม่ออก

2. **Transposition** วิธีการคือ นำข้อมูลมาทีละอักขระเรียงลงในอะเรย์สองมิติ (2-dimension array) ทีละคอลัมน์ (Column) โดยเริ่มจากแถว (Row) แรกก่อนและต่อไปทีละแถวจนจบข้อมูล จากนั้นจะใช้วิธีอ่านข้อมูลกลับมาทีละแถว เริ่มจากคอลัมน์แรกและอ่านไล่ต่อไปทีละคอลัมน์จนหมดอะเรย์

3. **Bit manipulation** จะทำได้โดยนำข้อมูลมาทีละอักขระแล้วใช้ Bit operator มากระทำกับข้อมูล

สำหรับโปรแกรมของแต่ละวิธีที่แสดงไว้นี้เขียนโดยใช้เทอร์โบซี (Turbo C ver 1.0) ซึ่งฟังก์ชัน Encode จะทำหน้าที่นำข้อมูลมาเข้ารหัส ส่วนฟังก์ชัน Decode จะเป็นตัวถอดรหัส

Substitution Ciphers

วิธีที่ง่ายที่สุดก็คือ การนำเอาข้อมูลแต่ละอักขระมาเปลี่ยนให้เป็นรหัสแอสกี (Ascii code) จากนั้นจะลบออก หรือบวกเข้าด้วยค่าคงที่หนึ่ง ซึ่งจะทำให้รหัสแอสกีของอักขระเดิมเปลี่ยนแปลงไปก็จะทำให้ข้อมูลอ่านไม่ออก เช่น ถ้าบวกเข้าด้วย 2 จะทำให้อักขระแต่ละตัวเปลี่ยนจากเดิมดังแสดงในรูปที่ 1

จากรูปที่ 2 จะแสดงวิธี Substitution ซึ่งถ้ารันโปรแกรมจะได้ผลดังนี้

เดิม	"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
กลายเป็น	"CDEFGHIJKLMNOPQRSTUVWXYZAB"

รูปที่ 1

รูปที่ 2 โปรแกรมแสดง Substitution Ciphers

```
#include <ctype.h>
#include <stdio.h>
void  encode(char *msg,char start)
{
      int  i = 0;
      char c;

      while ( ( c = *(msg+i) ) != NULL ) {
         if isalpha(c) {
            c += start;
            if isupper( *(msg+i) ) {
               if ( c > 'Z' ) c -= 26;
            }
```

รูปที่ 2 (ต่อ)

```
            else
               if ( c > 'z' ) c -= 26;
            *(msg+i) = c;
         }
         ++i;
      }
}
void  decode(char *msg,char start)
{
      int  i = 0;
      char c;

      while ( ( c = *(msg+i) ) != NULL ) {
         if isalpha(c) {
            c -= start;
            if isupper( *(msg+i) ) {
               if ( c < 'A' ) c += 26;
            }
            else
               if ( c < 'a' ) c += 26;
            *(msg+i) = c;
         }
         ++i;
      }
}
main()
{
      char  *data;

      data = "Kill him at the sea";
      printf("Before encode : %s\n",data);
      encode(data,2);
      printf("After  encode : %s\n",data);
      decode(data,2);
      printf("After  decode : %s\n",data);
}
```

```
Before encode : Kill him at the sea
After  encode : Mknn jko cv vjg ugc
After  decode : Kill him at the sea
```

แต่วิธีการนี้สำหรับการขโมยถอดรหัสทำได้ไม่ยากนัก เพราะตัวอักษรภาษาอังกฤษมีเพียง 26 ตัว ดังนั้นจึงสามารถทดลองเลือกค่าคงที่มาโดยมีค่าระหว่าง 1-26 ถ้าเราปรับปรุงโปรแกรมใหม่ตามรูปที่ 3 โดยสร้างอะเรย์ของอักขระขึ้นมาให้เรียงแบบสุ่ม ซึ่งในโปรแกรมรูปที่ 3 คือ c_encode และรวมเอาสเปซเข้ามาไว้ในอะเรย์ด้วย เพราะภาษาอังกฤษมีสเปซมาก จากนั้นก็เปลี่ยนวิธีการเข้ารหัสโดยการนำเอาอักขระแต่ละตัวของข้อมูลมาเปลี่ยนเป็น Index เพื่อใช้ดึงเอาอักขระตัวใหม่จากอะเรย์ของอักขระแบบสุ่มที่เราสร้างขึ้นมาแทนที่อักขระตัวเดิม วิธีการขโมยถอดรหัสทำได้ยากขึ้น เพราะเรามีวิธีเอาอักขระมาสร้างอะเรย์ได้ถึง 26! แบบ

ซึ่งถ้าทดลองรันโปรแกรมในรูปที่ 3 ก็จะได้ผลดังนี้

```
Before encode : Kill him at the sea
After  encode : FcvvPdctPqjPjdsPusq
After  dncode : Kill him at the sea
```

รูปที่ 3 โปรแกรมแสดง Substitution Ciphers โดยสร้างอะเรย์ของอักขระขึ้นมาให้เรียงแบบสุ่ม

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>
      char  c_encode[27] = "QAZWSXEDCRFVTGBYHNUJM IKOLP";
      char  c_decode[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ";
void  encode(char *msg)
{
      int   i = 0;
      char  c;

      while ( ( c = *(msg+i) ) != NULL ) {
         if ( isalpha(c) || isspace(c) ) {
            c = c_encode[strchr(c_decode,toupper(c))-c_decode];
            if islower( *(msg+i) ) c = tolower(c);
            *(msg+i) = c;
```

รูปที่ 3 (ต่อ)

```
            }
            ++i;
        }
    }
    void  decode(char *msg)
    {
        int   i = 0;
        char  c;

        while ( ( c = *(msg+i) ) != NULL ) {
            if ( isalpha(c) || isspace(c) ) {
                c = c_decode[strchr(c_encode,toupper(c))-c_encode];
                if islower( *(msg+i) ) c = tolower(c);
                *(msg+i) = c;
            }
            ++i;
        }
    }
    main()
    {
        char  *data;

        data = "Kill him at the sea";
        printf("Before encode : %s\n",data);
        encode(data);
        printf("After  encode : %s\n",data);
        decode(data);
        printf("After  decode : %s\n",data);
    }
```

รูปที่ 4 โปรแกรมแสดง Substitution Ciphers โดยทำการแทนค่าอักขระด้วยอะเรย์ของอักขระที่เรียงแบบสุ่มซ้ำสองครั้ง

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>
      char  c_encode_1[27] = "QAZWSXEDCRFVTGBYHNUJM IKOLP";
      char  c_encode_2[27] = "POI UYTREWQASDFGHJKLMNBVCXZ";
      char  c_decode[27]   = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ";
      char  count[27];
char  index(char c)
{
      return((isalpha(c)) ? toupper(c) - 'A' : 26);
}
void  encode(char *msg)
{
      int   i;
      char  c, change = 1;

      for (i = 0; i < 27; ++i) count[i] = 0;
      i = 0;
      while ( ( c = *(msg+i) ) != NULL ) {
          if ( isalpha(c) || isspace(c) )
              if (change) {
                  c = c_encode_1[strchr(c_decode,toupper(c))-c_decode];
                  ++count[index(c)];
              }
```

รูปที่ 4 (ต่อ)

```
            else {
               c = c_encode_2[strchr(c_decode,toupper(c))-c_decode];
               ++count[index(c)];
            }
         if (count[index(c)] == 2) {
            change = ((change) ? 0 : 1);
            count[index(c)] = 0;
         }
         if islower( *(msg+i) ) c = tolower(c);
         *(msg+i) = c;
         ++i;
      }
}
void  decode(char *msg)
{
      int   i;
      char  c, change = 1;

      for (i = 0; i < 27; ++i) count[i] = 0;
      i = 0;
      while ( ( c = *(msg+i) ) != NULL ) {
         if ( isalpha(c) || isspace(c) )
            if (change) {
               ++count[index(c)];
               c = c_decode[strchr(c_encode_1,toupper(c))-c_encode_1];
            }
            else {
               ++count[index(c)];
               c = c_decode[strchr(c_encode_2,toupper(c))-c_encode_2];
            }
         if (count[index(*(msg+i))] == 2) {
            change = ((change) ? 0 : 1);
            count[index(*(msg+i))] = 0;
         }
         if islower( *(msg+i) ) c = tolower(c);
         *(msg+i) = c;
         ++i;
      }
}
main()
{
      char  *data;

      data = "Kill him at the sea";
      printf("Before encode : %s\n",data);
      encode(data);
      printf("After  encode : %s\n",data);
      decode(data);
      printf("After  decode : %s\n",data);
}
```

รูปที่ 5 โปรแกรมแสดง Transposition Ciphers

```
#include <alloc.h>
#include <mem.h>
#include <stdio.h>
#include <string.h>
#define  col  5
#define  row  6
```

จากการปรับปรุงโปรแกรมตามรูปที่ 3 นี้ก็ยังสามารถขโมยถอดรหัสได้ไม่ยากนัก ซึ่งถ้าเราสังเกตจากผลการรันโปรแกรม สำหรับอักขระที่เป็นสเปซจะถูกแทนที่ด้วยตัว P แต่เนื่องจาก

รูปที่ 5 (ต่อ)

```
      char  *skytale;
void  encode(char *msg,char *skytale)
{
      char  i, j;

      for (j = 0; j < col; ++j)
         for (i = 0; i < row; ++i) {
            if ( i*col+j <= strlen(msg) )
               *(skytale+j*row+i) = *(msg+i*col+j);
            else
               continue;
         }
}
void  decode(char *skytale,char *msg)
{
      char  i, j;

      for (i = 0; i < row; ++i)
         for (j = 0; j < col; ++j) {
            if ( i*col+j <= strlen(msg) )
               *(msg+i*col+j) = *(skytale+j*row+i);
            else
               continue;
         }
}
main()
{
      char  *data,i;

      data = "Kill him at the sea";
      skytale = (char *) malloc(sizeof(char));
      for (i = 0; i < col*row; ++i)
         if (i < col*row)
            *(skytale+i) = '#';
         else
            *(skytale+i) = NULL;
      printf("Before encode : %s\n",data);
      encode(data,skytale);
      printf("After  encode : %s\n",skytale);
      decode(skytale,data);
      printf("After  decode : %s\n",data);
}
```

รูปที่ 6 โปรแกรมแสดง Bit-Manipulation Ciphers ใช้วิธีการ 1's complement โดยใช้ not operator

```
#include <stdio.h>
void  encode(char *msg)
{
      int   i = 0;
      char  c;

      while ( ( c = *(msg+i) ) != NULL ) {
         *(msg+i) = ~c;
         ++i;
      }
}
```

ภาษาอังกฤษส่วนใหญ่จะมีแต่สเปซ ดังนั้นถ้าข้อมูลยาวก็จะมีตัว P มากด้วย และอาศัย Frequency table ที่ได้จากการเก็บสถิติความถี่ของการใช้ตัวอักษรแต่ละตัวของคำศัพท์ในภาษาอังกฤษก็จะช่วยถอดรหัสของข้อมูลได้ ดังนั้นเราจะปรับปรุงโปรแกรมได้ตามรูปที่ 4 โดยทำการแทนค่าอักขระด้วยอะเรย์ของอักขระที่เรียงแบบสุ่มซ้ำสองครั้งโดยโปรแกรมจะใช้อะเรย์ c__encode__1 สำหรับแทนก่อนจากนั้นจะคอยนับการปรากฏตัวของอักขระ ถ้าอักขระตัวใดปรากฏตัวครบสองครั้งก็จะสลับไปใช้อะเรย์ c__encode__2 สำหรับแทนข้อมูลจนกว่าจะมีอักขระตัวใดถูกใช้ครบสองครั้ง โปรแกรมจึงจะสลับกลับไปใช้อะเรย์ c__encode__1 ใหม่

ซึ่งถ้าทดลองรันโปรแกรมในรูปที่ 4 ก็จะได้ผลดังนี้

```
Before encode : Kill him at the sea
After  encode : Mknn jko cv vjg ugc
After  decode : Kill him at the sea
```

Transposition Ciphers

จากโปรแกรมตามรูปที่ 5 ใช้อะเรย์สองมิติขนาด 5×6 ในการเก็บตัวอักษร โดยโปรแกรมจะอ่านอักขระมาทีละตัวเรียงลงในอะเรย์เริ่มจากคอลัมน์ 1-5 ของแถวที่ 1 จากนั้นก็จะเรียงต่อในแถวต่อไปทีละคอลัมน์จนหมดข้อมูล แล้วก็จะอ่านข้อมูลกลับโดยอ่านไล่ทีละแถวของแต่ละคอลัมน์ซึ่งถ้าทดลองรันโปรแกรมในรูปที่ 5 ก็จะได้ผลดังนี้

```
Before encode : Kill him at the sea
After  encode : Kht ##ii s##lmte##l
                ha## ae###
After  decode : Kill him at the sea
```

สำหรับวิธี Transposition อาจจะเริ่มเรียงในแต่ละแถวทีละคอลัมน์ แล้วเวลาอ่านข้อมูลกลับก็จะเริ่มอ่านแต่ละคอลัมน์ทีละแถวแทน